RIVAL PENALIZATION CONTROLLED COMPETITIVE LEARNING FOR DATA CLUSTERING WITH UNKNOWN CLUSTER NUMBER

Yiu-ming Cheung

Department of Computer Science
Hong Kong Baptist University, Hong Kong
ymc@comp.hkbu.edu.hk

ABSTRACT

Conventional clustering algorithms such as k-means (Forgy 1965, MacQueen 1967) need to know the exact cluster number k^* before performing data clustering. Otherwise, they will lead to a poor clustering performance. Unfortunately, it is often hard to determine k^* in advance in many practical problems. Under the circumstances, Xu et al. in 1993 proposed an approach named *Rival Penalized Competitive Learning* (RPCL) algorithm that can perform appropriate clustering without knowing the cluster number by automatically driving extra seed points far away from the input data set. Although RPCL has made great success in many applications, its performance is however much sensitive to the selection of the de-learning rate. To our best knowledge, there is still an open problem to guide this rate selection. In this paper, we further investigate RPCL with presenting a mechanism to dynamically control the rival-penalizing forces. Consequently, we give out a rival penalized controlled competitive learning (RPCCL) approach, which circumvents the selecting problem of the de-learning rate by always fixing it at the same value as the learning rate. In contrast, the RPCL cannot do that in the same way. The experiments have shown the outstanding performance of this algorithm in comparison with the RPCL.

1. INTRODUCTION

As a statistical tool, clustering analysis has been widely applied in a variety of scientific areas such as pattern recognition, image processing, information retrieval and biology analysis. In the literature, the k-means [5] is a typical clustering algorithm, which partitions the input data set $\{\mathbf{x}_t\}_{t=1}^N$ that generally forms k^* true clusters into k categories (also simply called *clusters* without further distinction) with each represented by its center. Although the k-means has been widely used due to its easy implementation, it exists a serious potential problem. That is, it needs to pre-assign the number k of clusters. Many experiments have shown that it can work well only when k is equal to k^*. However, in many practical situations, it is hard or becomes impossible to know the exact cluster number in advance. Under the circumstances, the k-means algorithm often leads to a poor clustering performance. In the past decades, some works have been done along two major directions. The first one is to provide a way to determine the cluster number by formulating the cluster number selection as the choice of component number in a finite mixture model. Consequently, there have been some criteria proposed for model selection, such as AIC [1, 2], CAIC [3] and SIC [7]. Often, these existing criteria may overestimate or underestimate the cluster number due to the difficulty of choosing an appropriate penalty function. In recent years, a number selection criterion developed from a unified statistical learning theory named Ying-Yang Machine has been proposed and experimentally verified in [8, 9], whose computing however is laborious. In contrast, the other direction is to develop new advanced algorithms that perform clustering without pre-deciding the exact cluster number. For example, the typical incremental clustering gradually increases the number k of clusters under the control of a threshold value, which however is hard to be decided as well as k. Another typical example is the RPCL algorithm [10] that for each input, not only the winner of the seed points is updated to adapt to the input, but also its rival (the second winner) is de-learned by a smaller learning rate (also called *de-learning rate* hereafter). Many experiments have shown that the RPCL can automatically select the correct cluster number by gradually driving extra seed points far away from the input data set, but its performance is much sensitive to the selection of the de-learning rate. To our best knowledge, it is still an open problem so far to guide this rate selection. To circumvent this problem, the sister paper [4] of this has proposed a new generalized k-means algorithm named k^**-means* that just updates the winner each time to adapt to the input, and meanwhile adjusts the priori probability of each cluster occurrence. Although it has explicitly discarded the de-learning rule like in the RPCL , it actually penalizes not only the rival each

The work described in this paper was supported by the Faculty Research Grant of Hong Kong Baptist University with project number: FRG/01-02/II-24.

time, but also all other competitors because of the summation constraint of the priori probabilities. The experiments in [4] have shown its success in data clustering.

In this paper, we will study a new alternative way to solve the selecting problem of the de-learning rate in RPCL. We have noticed that the RPCL [10] performs the rival penalization without considering the distance between the winner and the rival as given an input. Actually, the rival should be more penalized if its distance to the winner is closer than the one between the winner and the input. This idea is also consistent with the social scenario in our daily life. For example, in an election campaign, the competition between two candidates (we call the final winning person *the winner* and the other one *the rival*) will become more intense if their public opinion polls are closer. Otherwise, the winner will be almost sure to win the election with little penalizing the rival during the election campaign. Based on this idea, we therefore present a mechanism, in which the rival-penalized strength is dynamically adjusted based on the distance between the winner and the rival relative to the current input. We have associated this mechanism with the de-learning rule of the RPCL, whereby a new improved algorithm named *Rival Penalization Controlled Competitive Learning* (RPCCL) is proposed. Compared to the RPCL, this new one always fixes the de-learning rate at the same value as the learning rate without requesting further determination. Such a setting is however not allowed in the RPCL as pointed out in [10]. The experiments have shown that this algorithm can smoothly work in all cases we have tried so far, but the RPCL cannot guarantee working given a pre-assigned de-learning rate. Also, we found that the proposed algorithm often gives correct clustering results much faster than the RPCL because the former generally gives stronger rival penalization strength than the latter when the current clustering partition is not appropriate.

2. OVERVIEW OF RPCL ALGORITHM

Given a set of data $D = \{\mathbf{x}_t\}_{t=1}^N$ that forms k^* clusters, the RPCL algorithm [10] is to perform clustering by learning k seed points, denoted as $\{\mathbf{m}_j\}_{j=1}^{k^*}$, whereby a data point (also called *input* interchangeably) can be correctly classified into the j^{th} cluster if the indicator function $I(j|\mathbf{x}_t) = 1$, with

$$I(j|\mathbf{x}_t) = \begin{cases} 1, & \text{if } j = \arg\min_r \|\mathbf{x}_t - \mathbf{m}_r\|^2 \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The basic idea of RPCL is that for each input, not only the winner seed point is modified to adapt to the input, but also its rival (the 2nd winner) is de-learned by a smaller learning rate. Specifically, the algorithm is:

Step 1: Randomly take a sample $\mathbf{x}_t$ from the data set D, let

$$I(j|\mathbf{x}_t) = \begin{cases} 1, & \text{if } j = c, \\ -1, & \text{if } y = r, \\ 0, & \text{otherwise,} \end{cases} \qquad 1 \le j \le k \quad (2)$$

with

$$\begin{aligned} c &= \arg\min_j \gamma_j \|\mathbf{x}_t - \mathbf{m}_j\|^2, \\ r &= \arg\min_{j \neq c} \gamma_j \|\mathbf{x}_t - \mathbf{m}_j\|^2, \end{aligned} \quad (3)$$

where $\gamma_j = \frac{n_j}{\sum_{r=1}^k n_r}$ is the relative winning frequency of the seed point $\mathbf{m}_j$ in the past, and n_j is the cumulative number of the occurrences of $I(j|\mathbf{x}_t) = 1$ in the past.

Step 2: Update the winner $\mathbf{m}_c$ (i.e., $I(c|\mathbf{x}_t) = 1$) and its rival $\mathbf{m}_r$ only by

$$\mathbf{m}_\tau^{\text{new}} = \mathbf{m}_\tau^{\text{old}} + \Delta\mathbf{m}_\tau, \quad \tau = c, r \quad (4)$$

with

$$\begin{aligned} \Delta\mathbf{m}_c &= \alpha_c(\mathbf{x}_t - \mathbf{m}_c) \\ \Delta\mathbf{m}_r &= -\alpha_r(\mathbf{x}_t - \mathbf{m}_r), \end{aligned} \quad (5)$$

where α_c and α_r are both the small positive learning rate, and often set $\alpha_r << \alpha_c$ as shown in [10]. Hereafter, α_c is simply called the learning rate, whereas α_r is called *de-learning rate* upon the fact that the rival is penalized by Eq.(5).

The experimental results in [10] have shown that, so long as k is initialized larger than the true one k^*, the RPCL can perform correctly clustering by driving extra seed points far away from the input data set. However, many experiments have also found that the performance of RPCL is sensitive to the selection of the de-learning rate α_r. In general, α_r needs to be re-selected appropriately not only for different clustering problems, but also for different initial positions of the seed points. Unfortunately, such an appropriate re-selection is not easy to do. To our best knowledge, there has not been an efficient way to choose α_r so far.

3. RPCCL ALGORITHM

The RPCCL approach invokes a new mechanism to control the rival penalization. The underlying idea of this mechanism is that the rival should be fully penalized if its distance to the winner is closer than the distance between the winner and the input; otherwise the penalization strength will be gradually attenuated when the distance between the winner and the rival increases. Since α_r is generally smaller than α_c, we can regard $\alpha_r = \alpha_c$ as a kind of fully penalization.

In this way, we can realize such a penalization mechanism by $\alpha_c p_r(\mathbf{x}_t)$ with

$$p_r(\mathbf{x}_t) = \frac{\min(|\mathbf{m}_c - \mathbf{m}_r|, |\mathbf{m}_c - \mathbf{x}_t|)}{|\mathbf{m}_c - \mathbf{m}_r|}, \quad (6)$$

where we have used Euclidean distance to measure the distance between two seed points. It can be seen that as $|\mathbf{m}_c - \mathbf{m}_r| \leq |\mathbf{m}_c - \mathbf{x}_t|$, the rival will be fully penalized with the rate α_c. Otherwise, the rival will be penalized with the rate $\alpha_c \frac{|\mathbf{m}_c - \mathbf{x}_t|}{|\mathbf{m}_c - \mathbf{m}_r|}$, which is gradually attenuated as $|\mathbf{m}_c - \mathbf{m}_r|$ increases. Hereafter, we also call $\alpha_c p_r(\mathbf{x}_t)$ the *rival-penalization strength*. We then associate this mechanism into the RPCL learning rules as given in Eq.(5). Consequently, the detailed RPCCL algorithm is as follows:

step 1: Randomly take an input $\mathbf{x}_t$ from the data set $D = \{\mathbf{x}_t\}_{t=1}^N$, calculate $I(j|\mathbf{x}_t)$ by Eq.(2).

step 2: Update the winner $\mathbf{m}_c$ and the rival $\mathbf{m}_r$ only by Eq.(4), but Eq.(5) becomes

$$\begin{aligned} \Delta\mathbf{m}_c &= \alpha_c(\mathbf{x}_t - \mathbf{m}_c) \\ \Delta\mathbf{m}_r &= -\alpha_c p_r(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{m}_r). \end{aligned} \quad (7)$$

These two steps are repeated for each input until $I(j|\mathbf{x}_t)$'s converge.

It can be seen that if we fix $p_r(\mathbf{x}_t)$ at a value smaller than 1, Eq.(7) is then equal to Eq.(5) with $\alpha_r = \alpha_c p_r(\mathbf{x}_t)$. That is, RPCCL is actually a generalization of the RPCL with including it as a special case. Moreover, it should be noted that when RPCCL gives an unappropriate clustering at current time step, i.e., there are two or more seed points located in one true cluster, the rival penalization strength in Eq.(7) is not less than $0.25\alpha_c$ in average if the data are uniformly distributed in the cluster. Such a big rival penalization strength however can lead the RPCL to break down totally. Hence, the α_r in Eq.(5) is generally much smaller than $0.25\alpha_c$, resulting in the RPCCL driving the extra seed points far away from the clusters much faster than the RPCL in general.

In addition, please note that the value of $p_r(\mathbf{x}_t)$ in Eq.(6) is always between 0 and 1, which can be therefore regarded as the probability of rival penalization. As a result, we can alternatively give out a stochastic version of RPCCL named Stochastic RPCL (S-RPCL), whose algorithm is as follows:

Step 1: This step is the same as that of RPCCL.

Step 2: Update the winner $\mathbf{m}_c$ by Eq.(7). Then uniformly generate a random number $\nu \in [0, 1]$. Let

$$\varrho = \begin{cases} 1, & \text{if} \quad \nu \leq p_r(\mathbf{x}_t); \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

We update the rival $\mathbf{m}_r$ by

$$\Delta\mathbf{m}_r = -\alpha_c \varrho(\mathbf{x}_t - \mathbf{m}_r). \quad (9)$$

In Eq.(8), the value of ϱ is switched between 0 and 1, which makes the rival penalization in Eq.(9) is implemented discontinuously. Actually, it can be seen that Eq.(9) is exactly equal to the de-learning rule of the RPCL in Eq.(5) with the de-learning rate $\alpha_r = \alpha_c$ if we always set $\varrho = 1$. As pointed out in [10], we have known that the RPCL cannot work completely if $\alpha_r = \alpha_c$. But, one interesting thing is that the S-RPCL can work well by just letting the rival penalization be done discontinuously under this controlled way. Since the rival effects of S-RPCL as a whole are the same as the RPCCL, it is generally expected that the performance of S-RPCL is the same as RPCCL. In the following, due to the space limit, we will just show the performance of RPCCL in comparison with the RPCL.

4. EXPERIMENTAL RESULTS

To compare RPCCL with the RPCL, we conducted four experiments. In each one, we used a set of data points with the size $N = 1,000$. Furthermore, we used six seed points, whose initial positions are all randomly assigned in the input data space. In addition, we randomly set the learning rate $\alpha_c = 0.001$ while letting $\alpha_r = 0.0001$ by default for the RPCL.

4.1. Experiment 1

We used the $1,000$ data points from a mixture of three Gaussian distributions with the true means: $(1, 1)^T$, $(1, 5)^T$ and $(5, 5)^T$, respectively. As shown in Fig. 1(a), the data form three well-separated clusters with the six seed points $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_6$ randomly located at:

$$\begin{aligned} \mathbf{m}_1 &= \begin{pmatrix} 2.2580 \\ 1.9849 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 1.4659 \\ 5.1359 \end{pmatrix}, \\ \mathbf{m}_3 &= \begin{pmatrix} 0.6893 \\ 5.0331 \end{pmatrix}, \quad \mathbf{m}_4 = \begin{pmatrix} 5.2045 \\ 5.1298 \end{pmatrix}, \\ \mathbf{m}_5 &= \begin{pmatrix} 1.9193 \\ 5.4489 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 5.5869 \\ 5.1937 \end{pmatrix}. \end{aligned} \quad (10)$$

After 100 epoches (The $1,000$ data points are scanned once is called *an epoch*), the six seed points learned by RPCCL have been converged to:

$$\begin{aligned} \mathbf{m}_1 &= \begin{pmatrix} 1.0131 \\ 0.9806 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0.9845 \\ 4.9823 \end{pmatrix}, \\ \mathbf{m}_3 &= \begin{pmatrix} -3.5557 \\ 5.4466 \end{pmatrix}, \quad \mathbf{m}_4 = \begin{pmatrix} 5.0180 \\ 5.0043 \end{pmatrix}, \\ \mathbf{m}_5 &= \begin{pmatrix} 5.7498 \\ 25.0371 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 11.4483 \\ 7.2208 \end{pmatrix} \end{aligned} \quad (11)$$

As shown in Fig. 1(b), RPCCL put three seed points: $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_4$ into the appropriate positions of three clusters meanwhile driving the other three extra seed points:

$\mathbf{m}_3$, $\mathbf{m}_5$ and $\mathbf{m}_6$ far away from the input data set. That is, RPCCL has successfully worked in this case. However, after 100 epoches, RPCL just led the six seed points to:

$$\begin{aligned} \mathbf{m}_1 &= \begin{pmatrix} 1.0131 \\ 0.9806 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0.9862 \\ 5.1899 \end{pmatrix}, \\ \mathbf{m}_3 &= \begin{pmatrix} -0.2110 \\ 7.2922 \end{pmatrix}, \quad \mathbf{m}_4 = \begin{pmatrix} 4.7637 \\ 5.0947 \end{pmatrix}, \\ \mathbf{m}_5 &= \begin{pmatrix} 8.9795 \\ 42.2057 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 5.3665 \\ 4.9695 \end{pmatrix}. \end{aligned} \tag{12}$$

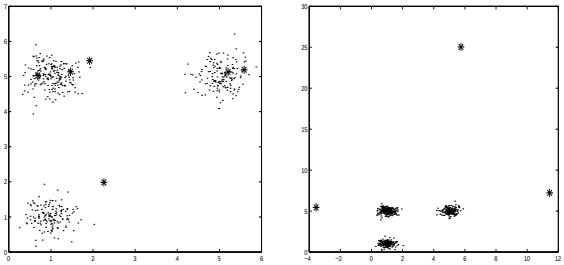
As shown in Figure 2(a), the RPCL drove only one seed point $\mathbf{m}_5$ far away from the input data set. We then further learned the seed points to the 200 epoch number. In this case, the RPCL has successfully driven three extra points $\mathbf{m}_3$, $\mathbf{m}_5$ and $\mathbf{m}_6$ to

$$\begin{aligned} \mathbf{m}_3 &= \begin{pmatrix} -3.0326 \\ 13.2891 \end{pmatrix}, \quad \mathbf{m}_5 = \begin{pmatrix} 14.4600 \\ 70.6014 \end{pmatrix}, \\ \mathbf{m}_6 &= \begin{pmatrix} 10.4714 \\ 5.2240 \end{pmatrix}, \end{aligned} \tag{13}$$

which are far away from the input data set as shown in Figure 2(b), while the other three seed points located at the correct positions as follows:

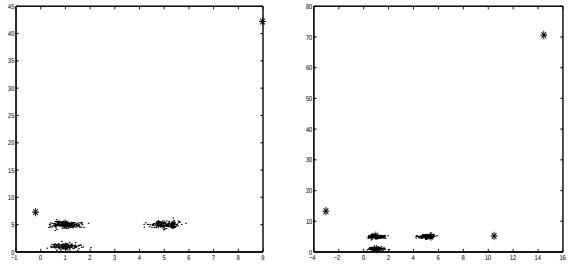
$$\begin{aligned} \mathbf{m}_1 &= \begin{pmatrix} 1.0167 \\ 0.9321 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0.9752 \\ 5.3068 \end{pmatrix}, \\ \mathbf{m}_4 &= \begin{pmatrix} 5.4022 \\ 5.0054 \end{pmatrix}. \end{aligned} \tag{14}$$

It should be noted that RPCL can work well in Experiment 1, but it needs more computing costs in comparison with RPCCL and S-RPCL.



(a) (b)

Figure 1: The positions of six seed points marked by '*' in the input data space at different steps in Experiment 1: (a) the initial positions, (b) the final position obtained via RPCCL.

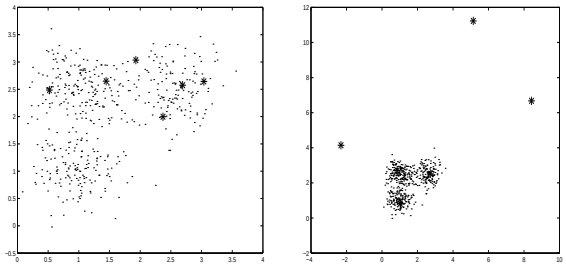


(a) (b)

Figure 2: (a)The experimental results of RPCL in Experiment 1. (a) The final position of six seed points obtained via RPCL with epoch = 100, where only one seed point is far away from the input set; (b) The final position obtained via RPCL with epoch = 200, where three extra seed points have been all driven away from the input set.

4.2. Experiment 2

Also, we used the $1,000$ data points from a mixture of another three Gaussian distributions with true means respectively: $(1,1)^T$, $(1,2.5)^T$ and $(2.5,2.5)^T$, which form three ball-shaped cluster but with serious overlapping area as shown in Figure 3(a) in comparison with the case in Experiment 1. After 100 epoches, we found that the RPCCL has given out the correct results as shown in Figure 3(b), but RPCL cannot work even if we increase the epoch number to $1,000$. Also, we further tested RPCL by adjusting α_r from 0.0009 to 0.00001 with a constant decreased step: 0.00001. Unfortunately, we could not find out an appropriate α_r in all cases we have tried so far to let RPCL successfully work.



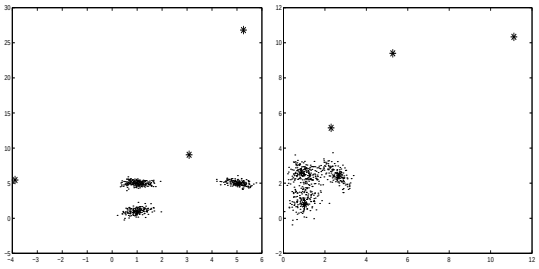
(a) (b)

Figure 3: The positions of six seed points marked by '*' in the input data space at different steps in Experiment 2: (a) the initial positions, (b) the final position obtained via RPCCL.

4.3. Experiment 3 and 4

In the previous two experiments, we assume that each cluster is ball-shaped, but which is not always true in many practical clustering problems, where the data may form ellipse-shaped clusters. Although we can modify the updating rule of the seed points in Eq.(7) by considering the data covariance matrices. Here, we prefer to the simple rule structure, and we would show the robust performance of RPCCL in two more general situations.

Similarly to Experiment 1 and Experiment 2, Experiment 3 considers the ellipse-shaped clusters with well separated, whereas Experiment 4 deals with the case with data clusters seriously overlapped. Again, under the same experimental environment, we performed the clustering by using RPCCL. Figure 4(a) and Figure 4(b) all showed that the RPCCL have successfully drove the three extra seed points far away from the input data set, meanwhile locating the other three seed points at the correct positions.



(a) (b)

Figure 4: The final positions of six seed points marked by '*' in the input data space, obtained by (a) Experiment 3; (b) Experiment 4.

5. CONCLUSION

We have further investigated the RPCL with presenting a mechanism to dynamically control the rival-penalizing forces. Consequently, we have given out the RPCCL algorithm and its stochastic version. Compared to the RPCL, the main advantage of the proposed algorithm is that it need not determine the value of de-learning rate, while performing correct clustering without knowing exact cluster number. The experiments have demonstrated its outstanding performance.

6. REFERENCES

[1] H. Akaike, "Information Theory and An Extension of the Maximum Likelihood Principle", *Proceedings of Second International Symposium on Information Theory*, pp. 267–281, 1973.

[2] H. Akaike, "A New Look at the Statistical Model Identfication", *IEEE Transactions on Automatic Control AC-19*, pp. 716–723, 1974.

[3] H. Bozdogan, "Model Selection and Akaike's Information Criterion: The General Theory and its Analytical Extensions", *Psychometrika*, Vol. 52, No. 3, pp. 345–370, 1987.

[4] Y.M. Cheung, "k^*-means — A Generalized k-means Clustering Algorithm with Unknown Cluster Number", *Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning* (IDEAL'02), pp. 307–317, August 12-14, Manchester, UK, 2002.

[5] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations", *Proceedings of* 5^{nd} *Berkeley Symposium on Mathematical Statistics and Probability*, 1, Berkeley, Calif.: University of California Press, pp. 281–297, 1967.

[6] E.W. Forgy, "Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications", *Biometric Soc. Meetings*, Riverside, California (Abstract in Biometrics 21, No. 3, 768), 1965.

[7] G. Schwarz, "Estimating the Dimension of a Model", *The Annals of Statistics*, Vol. 6, No. 2, pp. 461–464, 1978.

[8] L. Xu, "How Many Clusters?: A Ying-Yang Machine Based Theory for A Classical Open Problem in Pattern Recognition", *Proceedings of IEEE International Conference on Neural Networks*, Vol. 3, pp. 1546–1551, 1996.

[9] L. Xu, "Bayesian Ying-Yang Machine, Clustering and Number of Clusters", *Pattern Recognition Letters*, Vol. 18, No. 11-13, pp. 1167–1178, 1997.

[10] L. Xu, A. Krzyżak and E. Oja, "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection", *IEEE Transaction on Neural Networks*, Vol. 4, pp. 636–648, 1993. Its preliminary version was appeared in *Proceedings of 1992 International Joint Conference on Neural Networks*, Vol. 2, pp. 665–670, 1992.